TITLE OF THE PROJECT:

Designing and implementing a 3-bit up/down counter using Verilog.

Team Members:

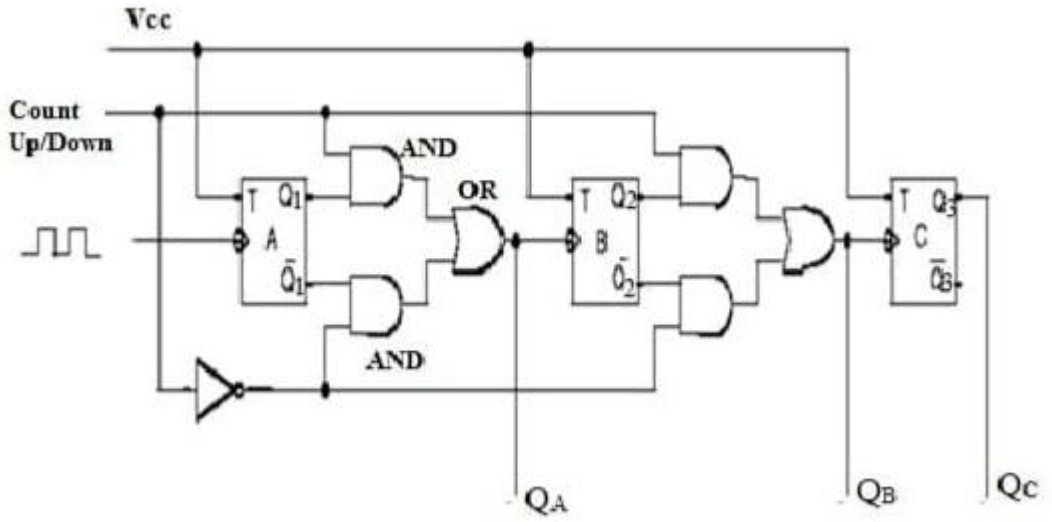
Sl. No	Name	SRN
1	Anurag Senapati	PES1UG23AM059
2	Anjali HR	PES1UG23AM053
3	Ahana Shetty	PES1UG23AM030

Explanation about the project in brief:

A 3-bit up/down counter in Verilog is a digital circuit that counts either up or down in a binary sequence, controlled by an up/down signal.
Key Components

1. 3-bit Counter: Since it's 3-bit, the counter can hold values from 000 (0 in decimal) to 111 (7 in decimal).
2. Up/Down Control Signal: This is a control input, typically a single bit (up_down) which determines whether the counter counts up (up_down = 1) or down (up_down = 0).
3. Clock Signal: The counting is synchronous and depends on a clock signal (clk), incrementing or decrementing on each clock cycle.
4. Reset Signal: A reset signal is used to reset the counter back to zero.

Circuit Diagram:



Iverilog code:

1. Main File:

```
module t_flip_flop (
  input t, clk, reset,
  output reg q
);
  always @(posedge clk or posedge reset) begin
    if (reset)
      q <= 0;
    else if (t)
      q <= ~q;
  end
endmodule

module up_down_counter (
  input clk, reset, up,
  output [2:0] count
);
  wire t0, t1, t2;
  wire q0, q1, q2;

  // T flip-flop logic for the counter
```

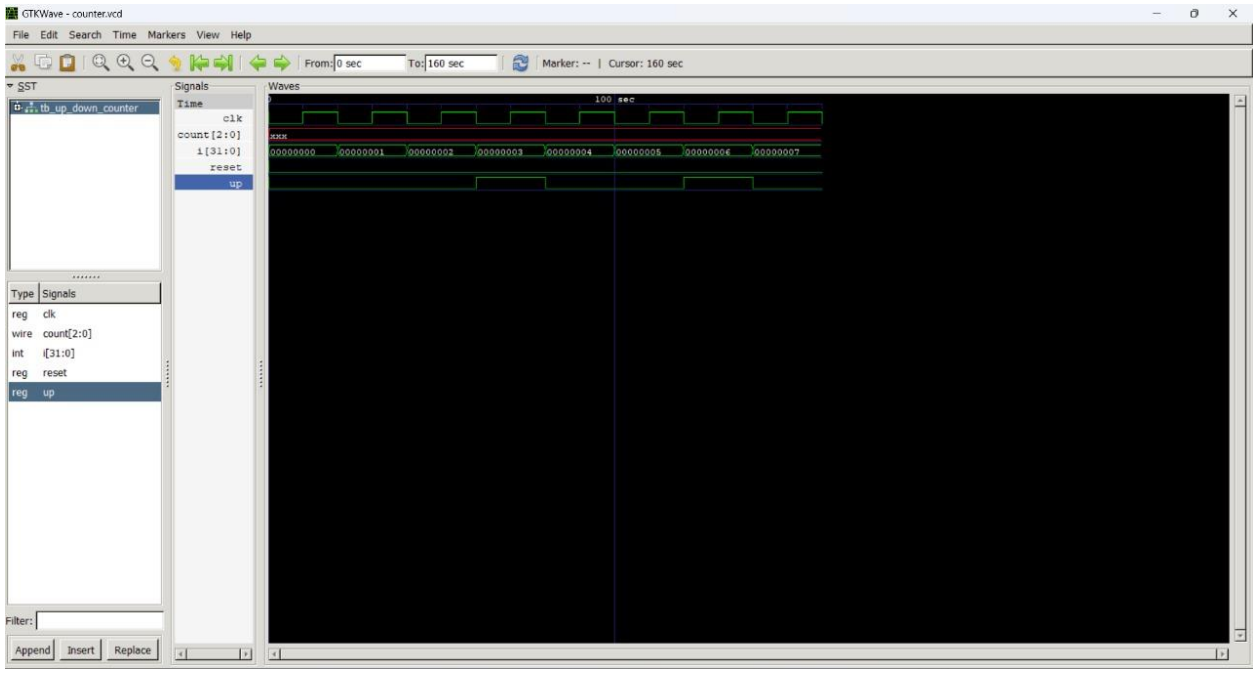
```
  assign t0 = 1;  // LSB toggles on every clock cycle
  assign t1 = (up) ? q0 : ~q0;  // Middle bit toggle logic based on up/down
  assign t2 = (up) ? (q0 & q1) : (~q0 & ~q1);  // MSB toggle logic based on up/down

  // Instantiate T flip-flops
  t_flip_flop ff0 (.t(t0), .clk(clk), .reset(reset), .q(q0));
  t_flip_flop ff1 (.t(t1), .clk(clk), .reset(reset), .q(q1));
  t_flip_flop ff2 (.t(t2), .clk(clk), .reset(reset), .q(q2));

  // Combine flip-flop outputs for the 3-bit count output
  assign count = {q2, q1, q0};
endmodule
```

2. Testbench File

```
`define TESTVECS 7
module tb_up_down_counter;
  reg clk;
  reg reset;
  reg up;
  wire [2:0] count;

  // Define the test vectors
  reg [2:0] test_vecs [0:(`TESTVECS-1)];
  integer i;

  // Instantiate the up_down_counter
  up_down_counter uut (
    .clk(clk),
    .reset(reset),
    .up(up),
    .count(count)
  );

  // Generate clock signal
  always #10 clk = ~clk;

  initial begin
    $dumpfile("counter.vcd");
    $dumpvars(0, tb_up_down_counter);
```

```
    // Initialize clock
    clk = 0;

    // Define the test vectors (reset, up)
    test_vecs[0] = 3'b100;  // reset = 1, up = 0 (initialize with reset)
    test_vecs[1] = 3'b000;  // reset = 0, up = 0 (count down)
    test_vecs[2] = 3'b001;  // reset = 0, up = 1 (count up)
    test_vecs[3] = 3'b000;  // reset = 0, up = 0 (count down)
    test_vecs[4] = 3'b100;  // reset = 1, up = 0 (reset while counting down)
    test_vecs[5] = 3'b001;  // reset = 0, up = 1 (count up after reset)
    test_vecs[6] = 3'b000;  // reset = 0, up = 0 (count down)

    // Initialize reset and up values
    {reset, up} = 0;

    // Apply test vectors
    for (i = 0; i < `TESTVECS; i = i + 1) begin
      #20 {reset, up} = test_vecs[i];
    end

    // Finish simulation
    #20 $finish;
  end
endmodule
```

3. GTKWave Output


GTKWave - counter.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 160 sec
Marker: -- | Cursor: 160 sec
SST
tb_up_down_counter
Signals
Time
clk
count[2:0]
i[31:0]
reset
up
Waves
100 sec
xxx
00000000
00000001
00000002
00000003
00000004
00000005
00000006
00000007
Type Signals
reg clk
wire count[2:0]
int i[31:0]
reg reset
reg up
Filter:
Append
Insert
Replace